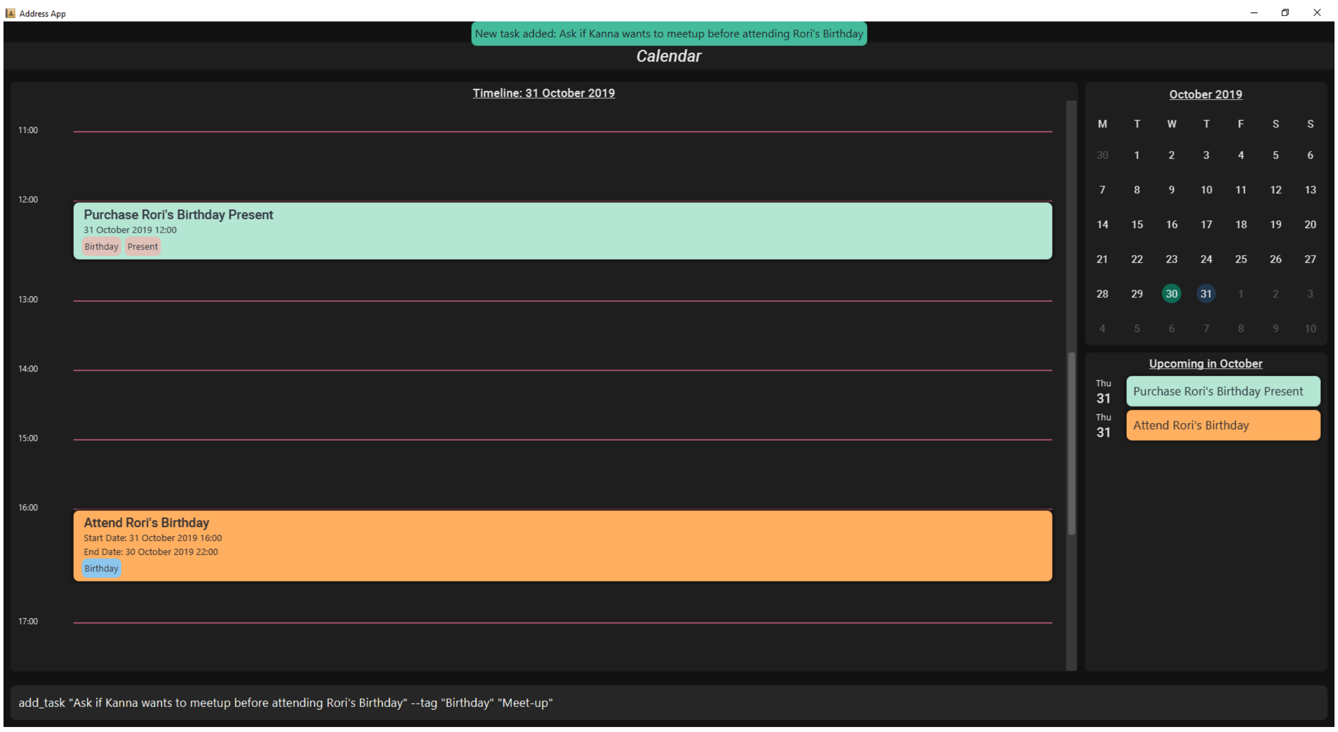
Timothy Yu Zhiwen - Project Portfolio for Horo

1. Overview

Tasked to enhance or morph a simple command line interface (CLI) desktop address book application, our team of 5 Computer Science Students decided to morph the program into calendar scheduling application for students. From the word "Horology", we came up with the name **Horo** for our program. It is capable of maintaining a calendar and to-do list, posting timely reminders for the users.

Here is a screenshot of **Horo**



New task added: Ask if Kanna wants to meetup before attending Rori's Birthday

Purchase Rori's Birthday Present
31 October 2019 12:00
Birthday Present

Attend Rori's Birthday
Start Date: 31 October 2019 16:00
End Date: 30 October 2019 22:00
Birthday

```
add_task "Ask if Kanna wants to meetup before attending Rori's Birthday" --tag "Birthday" "Meet-up"
```

Figure 1. Screenshot of Horo

I played the role for designing the User Interface (UI), and the logic of how my teammates' code will interact with the UI. The following sections below will display the following morphing I did in details, including certain documentation added to the User and Developer Guide.

Note: The following symbols and format are explained as:

Table 1. Symbol and Formats

`list`	A bold text with highlight as well as change to monospaced indicates a command input by you which is to be executed by the Horo.
`TimelineView`	A text with highlight as well as monospaced font indicates a component, class or object in the structure of **Horo**

2. Summary of contributions

2.1. Enhancement Ui and Calendar Logic

2.1.1. What is it:

There are 3 different UI screens/panels you can switch between: Calendar Panel, List Panel, Log Panel. Each of the panels shows you a different screen provided with different information.

2.1.2. What does it do:

This is accessed using the commands calendar, list, log respectively and provides a different screen to you depending on what you wish to see.

2.1.3. Why it is necessary:

It provides you a Graphical User Interface (GUI) so you can easily view their Events or Tasks with a simple switch between the views.

2.2. Code Contributed

2.2.1. Sample Code Contributed:

Here is a `Repo Sense Link` that indicates the amount of code I have written for the project. For more details, the most of the following code are also written by me, and authorship - `//@@author Kyzure`, or `<!-- @@author Kyzure -→` - can be found on the top of the code.

- `Ui Controller Code` - The packages `systemTray` and `listeners` are not written by me, as well as code without my authorship.
- `JavaFX Code for .fxml files`
- `Calendar Date Model Code`
- `Command Code` - Only the following is written by me: `DayViewCommand`, `WeekViewCommand`, `MonthViewCommand`, `CalendarViewCommand`, `ListViewCommand`, `LogViewCommand`.

2.3. Other Contribution

2.3.1. Documentation:

- Provided baseline for both User and Developer Guide at the beginning, and improve the design to make it more readable for the user. (Pull requests #11, #94, #95, #119, #128, #226)

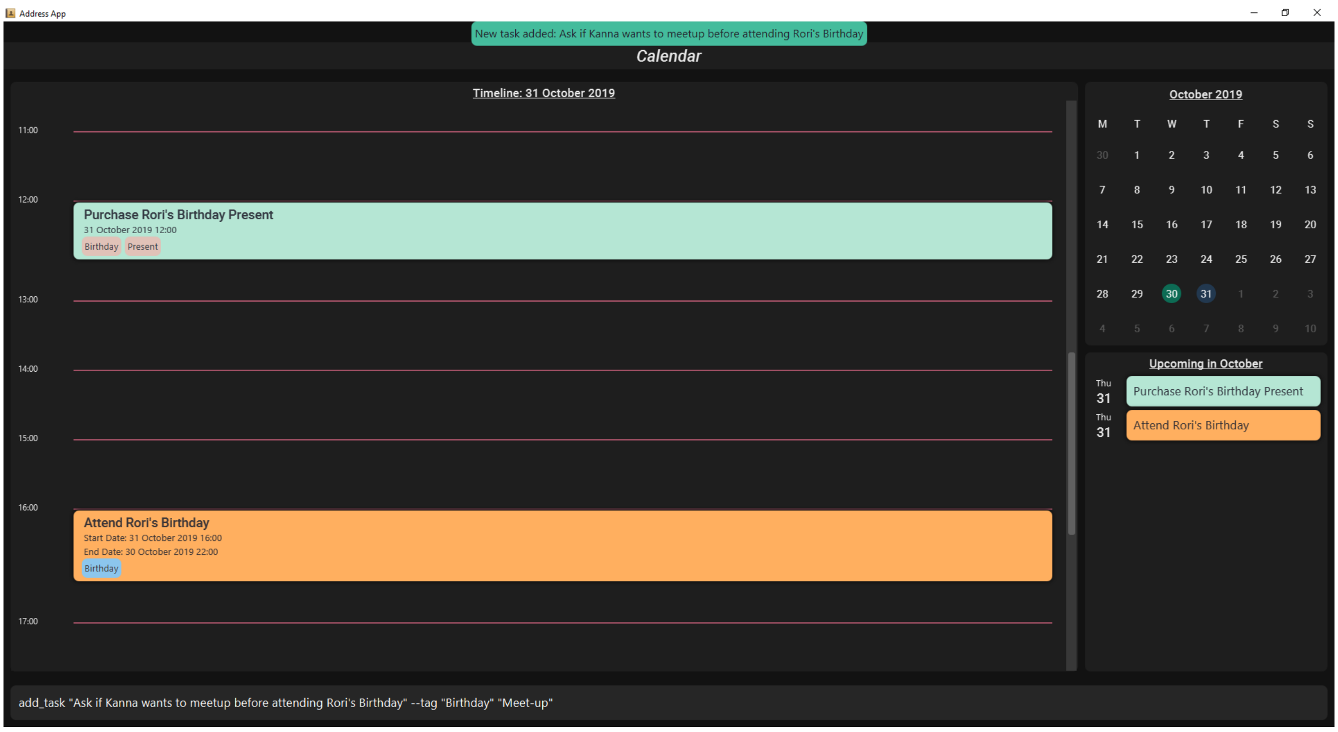
2.3.2. Community:

- Reviewed Pull Requests: #41

3. Contributions to the User Guide

The following sections are an excerpt of my contribution to the User Guide.

3.1. Changing Screen View to Calendar View



New task added: Ask if Kanna wants to meetup before attending Rori's Birthday

Purchase Rori's Birthday Present
31 October 2019 12:00
Birthday Present

Attend Rori's Birthday
Start Date: 31 October 2019 16:00
End Date: 30 October 2019 22:00
Birthday

Thu 31 Purchase Rori's Birthday Present
Thu 31 Attend Rori's Birthday

```
add_task "Ask if Kanna wants to meetup before attending Rori's Birthday" --tag "Birthday" "Meet-up"
```

Figure 2. Calendar View Command for Calendar

The `calendar` command switches the display to the Calendar View, which displays a calendar of the specified month and year in addition to a timeline of the specified day, week or month.

The Calendar View will display the specified date. If no date is specified, the last specified date will be displayed. This defaults to the current date.

Upon the initial launch of the application, the timeline and calendar dates will be set to the system's current date.

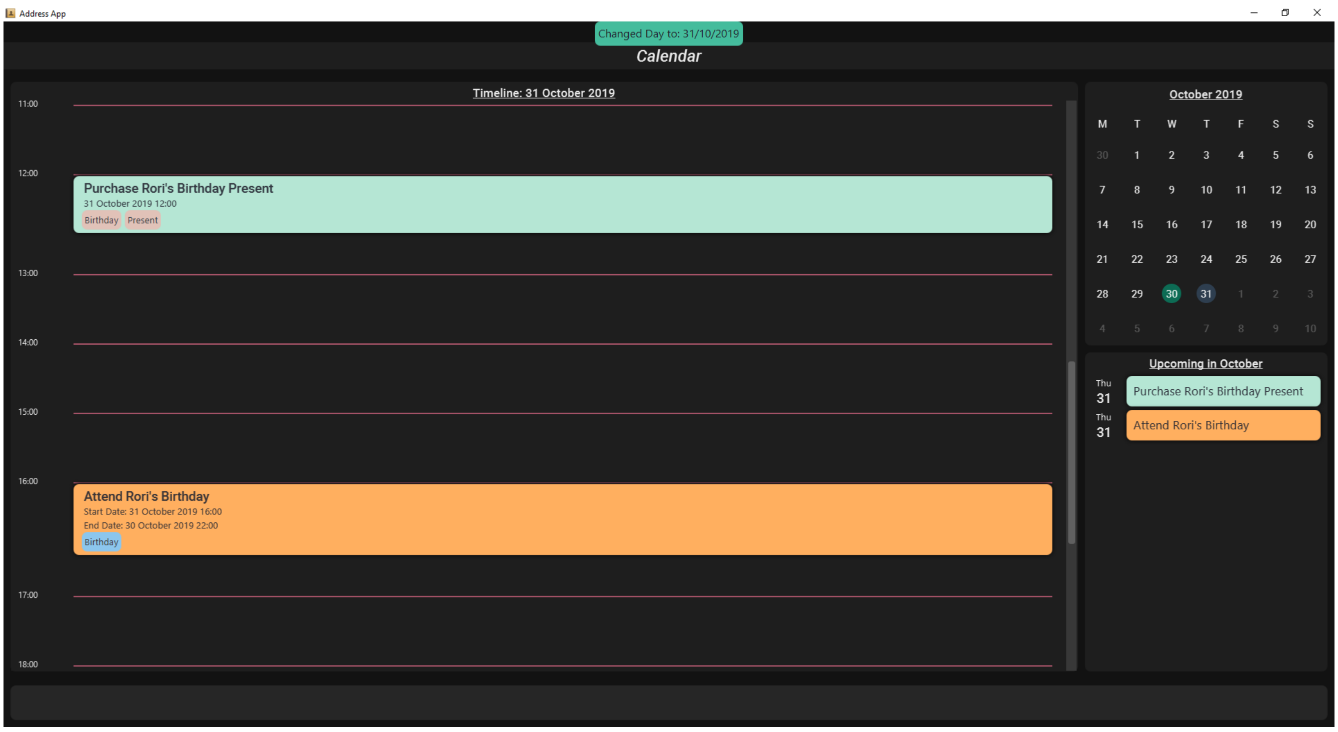
Command Format:
`calendar`

Command Parameters:
`--date MONTH_YEAR`

Argument Format:
`MONTH_YEAR` : `MM/YYYY`

Example:
`calendar` : Switch back to calendar view, without changing the date.
`calendar --date 10/2019`

3.2. Changing Timeline to a given day



Changed Day to: 31/10/2019

Purchase Rori's Birthday Present
31 October 2019 12:00
Birthday Present

Attend Rori's Birthday
Start Date: 31 October 2019 16:00
End Date: 30 October 2019 22:00
Birthday

Figure 3. Day View Command for the Timeline

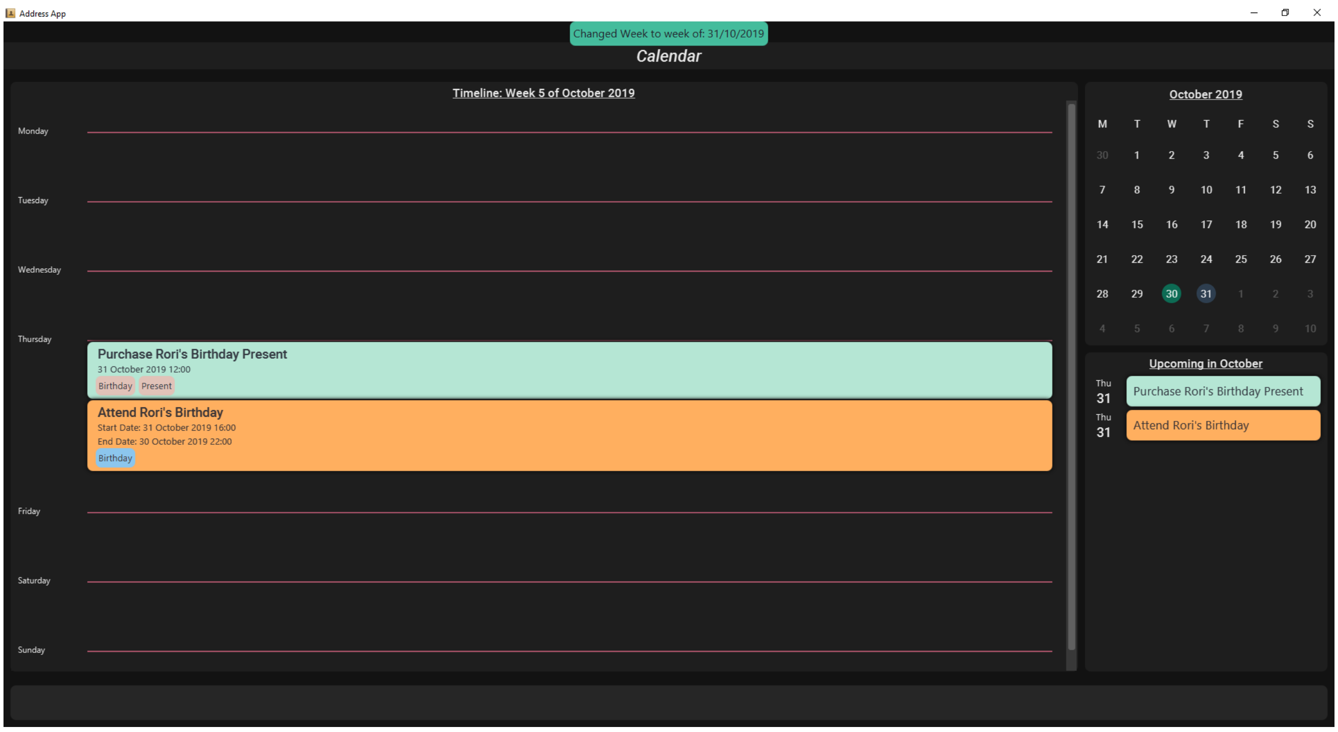
The `day` command sets the timeline in the Calendar View to that of the specified day. In addition, this command will cause a switch to the Calendar View if it is not the current display.

Command Format:
`day DATE`

Argument Format:
`DATE` : `DD/MM/YYYY`

Example:
`day 11/10/2019`

3.3. Changing Timeline to a given week



Changed Week to week of: 31/10/2019

Purchase Rori's Birthday Present
31 October 2019 12:00
Birthday Present

Attend Rori's Birthday
Start Date: 31 October 2019 16:00
End Date: 30 October 2019 22:00
Birthday

Figure 4. Week View Command for the Timeline

The `week` command sets the timeline in the Calendar View to the week of the specified day of the month. In addition, this command will cause a switch to the Calendar View if it is not the current display.

Command Format:
`week DATE`

Argument Format:
`DATE` : `DD/MM/YYYY`

Example:
`week 01/11/2019`

4. Contributions to the Developer Guide

The following sections are an excerpt of my contribution to the Developer Guide.

4.1. Implementation when changing the date of timeline

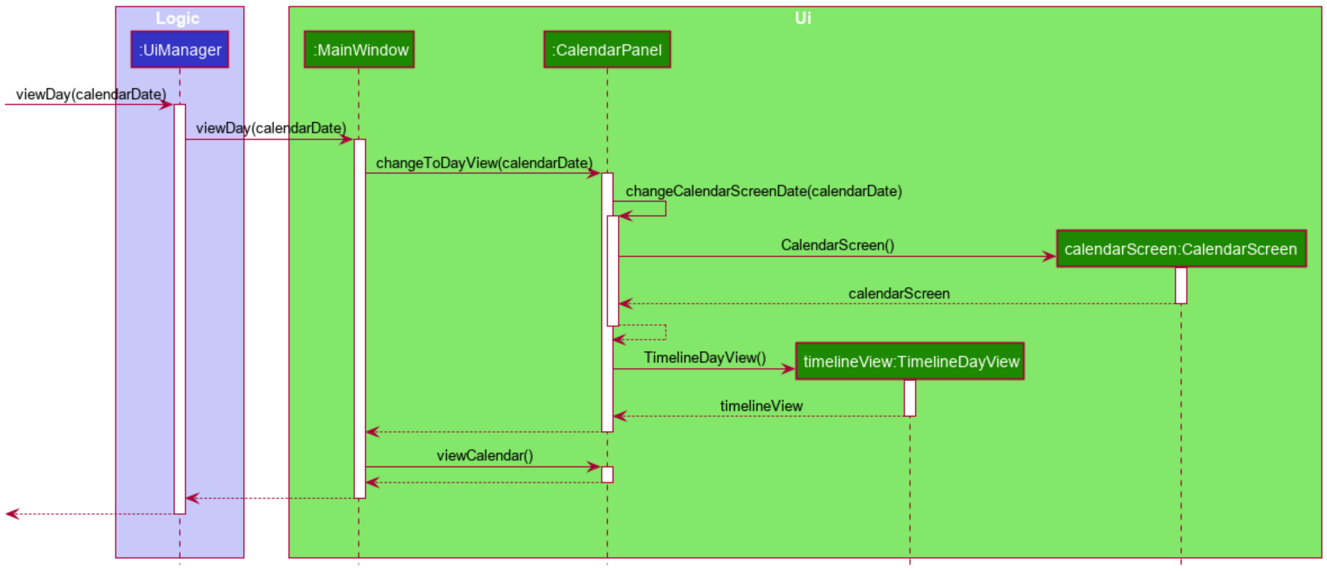


Figure 5. Sequence Diagram for changing the timeline date

Here is an example of the sequence for the UI when `DayViewCommand` is executed to change the date of the timeline.

Step 1. When the command is executed, it will proceed to call `UiManager` through `viewDay(calendarDate)`, which in turn will call `MainWindow` and subsequently `CalendarPanel`.

Step 2. `CalendarPanel` will proceed to execute `changeCalendarScreenDate(calendarDate)`, which will create an instance of `CalendarScreen` to display the calendar.

Step 3. Afterwards, a new instance of `TimelineDayView` will be created to display the timeline.

Step 4. Lastly, `MainWindow` will call `viewCalendar` which will be explained in the next section, so as to allow `CalendarPanel` to be visible while the other panels are not.

4.2. Implementation when changing views

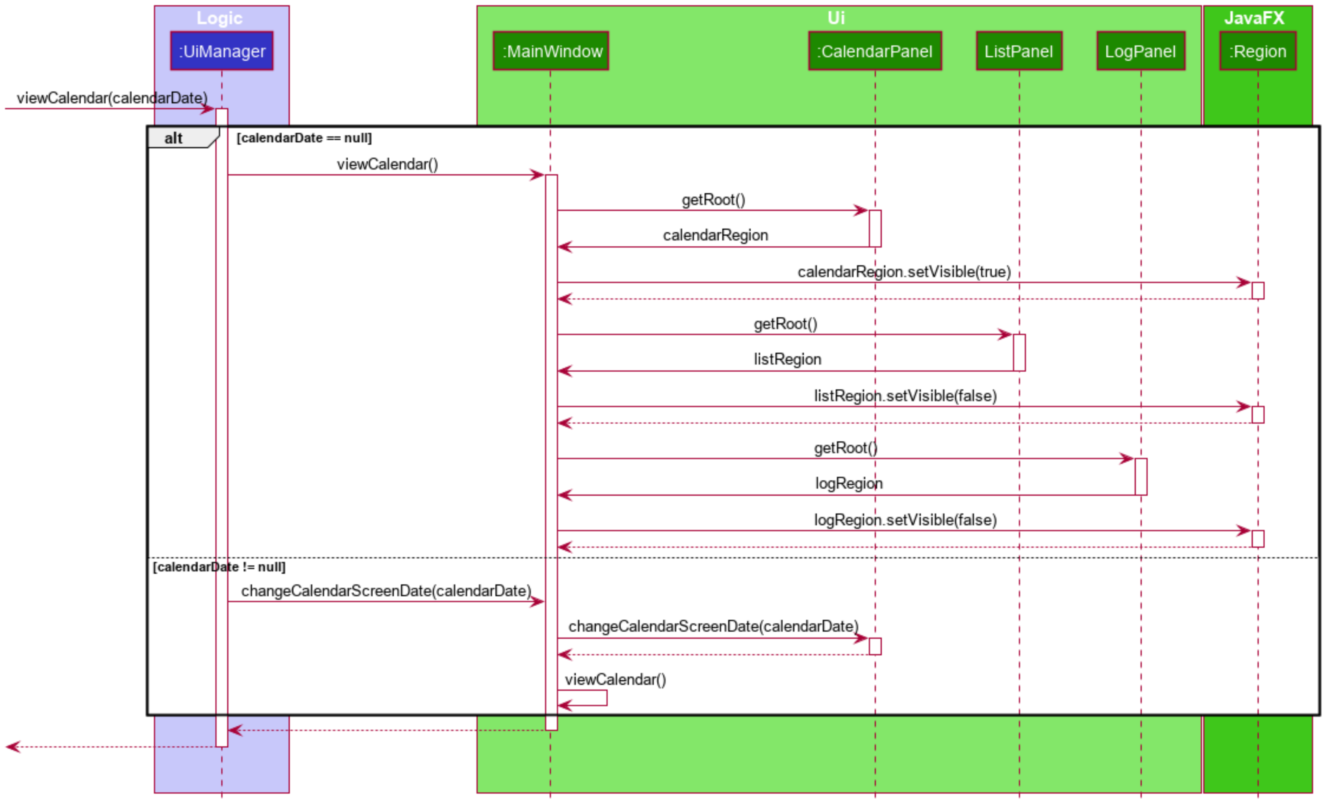


Figure 6. Sequence Diagram for changing to Calendar View

Here is an example of the sequence for the UI when `CalendarViewCommand` is executed.

Step 1. When command is executed, it will proceed to call `UiManager` through `viewCalendar(calendarDate)`, which will proceed to check if the giving date is null, or a date. The validity check is previously check in the parser.

Step 2. If calendarDate is null, the `UiManager` will simply call `MainWindow` to switch the view with the method `viewCalendar()`.

Step 3. `MainWindow` will obtain the `Region` of the 3 panels: `CalendarPanel`, `ListPanel` and `LogPanel`, and proceed to set only `CalendarPanel` to be visible.

Step 4. If calendarDate is not null, `UiManager` will then call `MainWindow` using `changeCalendarScreenDate(calendarDate)`, to change the `CalendarScreen` to the given date.

Step 5. Afterwards, it will proceed to repeat step 3 again, which is simply calling `viewCalendar()` again.

Since the sequence for `CalendarViewCommand` is roughly similar, or in fact, more complicated than `ListViewCommand` and `LogViewCommand`, it will not be further explained.

4.3. Design Considerations

The design considerations are more towards how the UI would have look like, as well as how the architecture of the code would have change if depending on the arrangement of the UI.

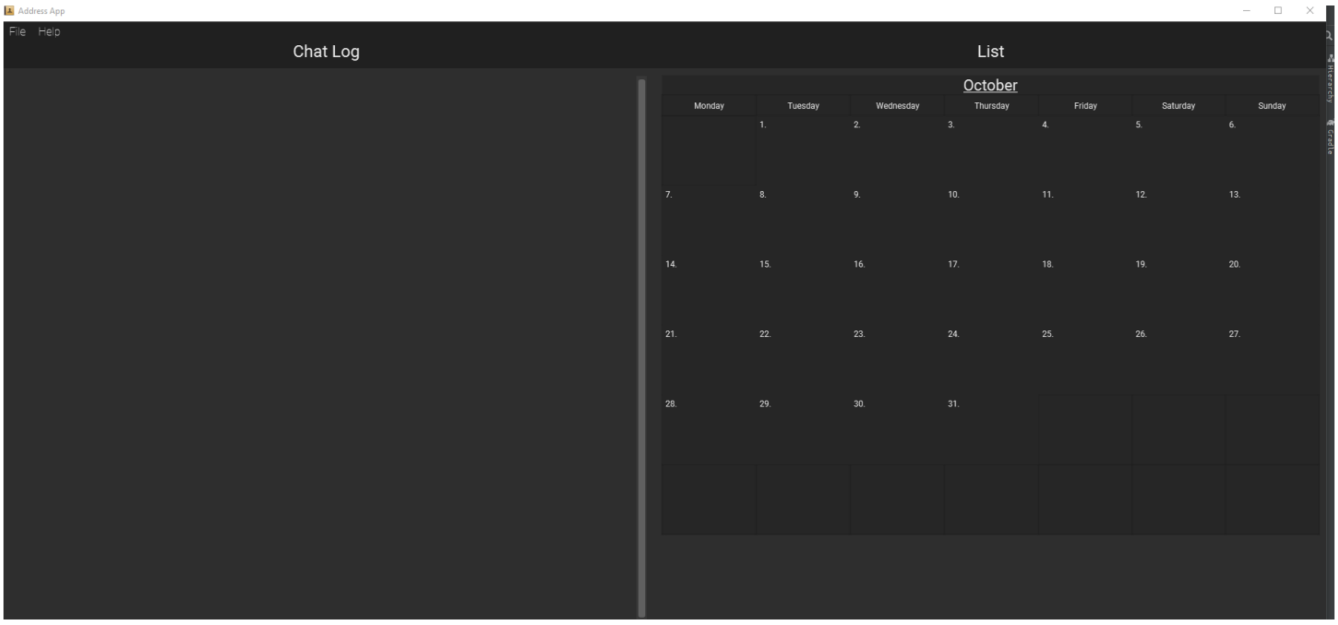
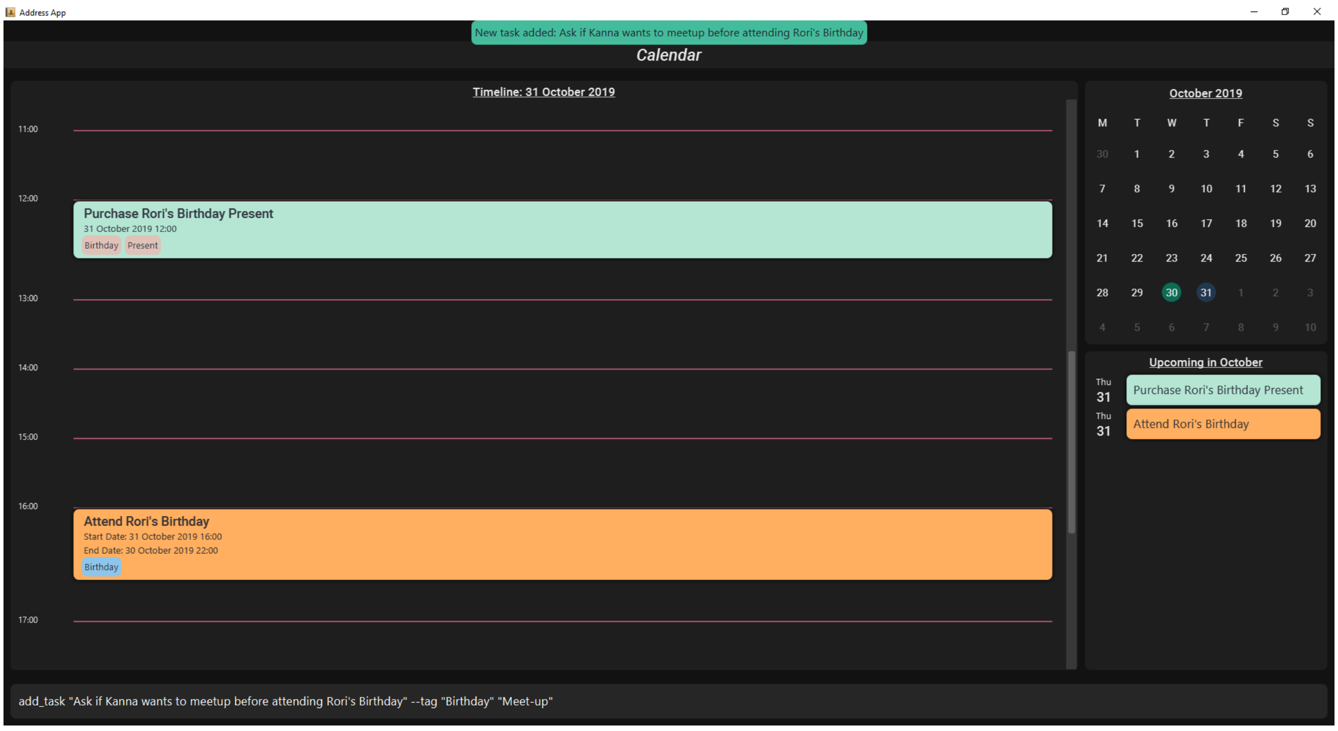


Figure 7. Old design of the UI



New task added: Ask if Kanna wants to meetup before attending Rori's Birthday

Purchase Rori's Birthday Present
31 October 2019 12:00
Birthday Present

Attend Rori's Birthday
Start Date: 31 October 2019 16:00
End Date: 30 October 2019 22:00
Birthday

Thu 31 Purchase Rori's Birthday Present
Thu 31 Attend Rori's Birthday

```
add_task "Ask if Kanna wants to meetup before attending Rori's Birthday" --tag "Birthday" "Meet-up"
```

Figure 8. Current design of the UI

Aspect: Design of the `CalendarPanel`

- Alternative 1: The `CalendarPanel` is of an actual calendar, depicting a number events and tasks on each day of the month.
  - **Pros**: It will provide a better representation of a calendar, allowing people to judge how much is going on in a day of that month in one look.
  - **Cons**: Due to the nature of how limited in size a calendar can be, the user will be required to either check `ListPanel` for the details of an event or tasks, or have an extra screen beside the calendar for the user to check the details.
  - **Cons**: Similarly, a calendar can only input up to a fix amount of events or tasks there are on a particular day.

- Alternative 2 (current choice): The `CalendarPanel` consists of a mini-calendar as well as a timeline. An additional slot for upcoming events and tasks was later designed with the increase in space.
  - **Pros**: There is a much more greater space to show how much events or tasks one can have in a day, week or month.
  - **Pros**: The user can easily managed and check the Events and Tasks of a certain day.
  - **Cons**: Even though it is a timeline, it is still rather similar to list view, just with the timeline added to limit the amount of events seen on that day, week or month.
  - **Cons**: :The user will not be able to know what Events or Tasks there are easily, unless they change the view to month view. However, the increase space allows a small section for the upcoming events and tasks which tackles this problem.

Aspect: Design of the `LogPanel`

- Alternative 1: The `LogPanel` is placed side-by-side with any other panel.
  - **Pros**: The users can always have a visualization of the success of their commands
  - **Cons**: A large portion of the space is used for the `LogPane`, even if the it is scaled down compared to the other panels.
  - **Cons**: Appearance-wise, it looks extremely clunky due to most of the users' time will be looking at the calendar or list itself instead of the log.
- Alternative 2 (current choice): The `LogPanel` is placed separately as a different panel which can be access at any time from other panels. After each command is typed, a pop-up box will appear to indicate the success or failure of the command.
  - **Pros**: Since the user most of the time would only want to know if their command is successful or not, the pop-up box will be sufficient for such indication.
  - **Cons**: The user will have to check the `LogPanel`

The initial design is as of the image above showing the old UI. However, we decided to scrape it and did an overhaul of the UI using alternative 2 instead. This is due to our decision of wanting a better-looking and minimalist UI instead of one packed with information.